Empathetic Processing in Cybersecurity: A Human-Centric Data Analytics Paradigm

Charles D. Herring

November 20, 2025

Abstract

Security Operations Centers (SOCs) today face an overwhelming flood of security alerts – on the order of **thousands per day** – yet struggle to derive actionable intelligence, with studies showing that over two-thirds of alerts go unattended and the majority are false positives [1]. *Empathetic Processing (EP)* is a proposed paradigm that addresses this "data overload, insight scarcity" problem by modeling the analytics pipeline on human communication. In an EP framework, AI-driven systems **"listen"** to diverse data sources with deeper comprehension (using NLP to interpret intent and context), **resolve dissonance** by correlating and reconciling events over time (using graph-based link analysis to connect related evidence), and **"speak"** their findings in narratives tailored to human stakeholders (translating raw data into investigation timelines, executive summaries, and compliance reports). Key innovations enabling EP include the **predestination of data** – structuring and labeling data at the moment of ingestion with all future analytic and compliance needs in mind – and **temporal link analysis** – constructing a time-sequenced graph of entities and events to support long-horizon correlations and hypothesis testing (akin to mapping an adversary's kill-chain [4]). By integrating natural language processing, expert system rules, and machine learning over a unified knowledge graph, Empathetic Processing emulates the intuition of an expert analyst at machine speed. This paper explores the design principles of EP and details how the WitFoo cybersecurity platform implements these ideas to achieve significant reductions in alert noise, faster incident resolution, and improved forensic readiness. We evaluate EP's impact on SOC efficiency and accuracy and discuss future research directions (such as leveraging large language models) that build on this human-centric analytics approach.

Introduction

Modern SOCs collect a **vast volume of security telemetry** – network logs, endpoint events, alerts from dozens of tools – but converting this raw data into useful insight is increasingly challenging. Analysts are inundated by more alerts than they can possibly

investigate. A recent global study of 2,000 SOC analysts found that teams receive **4,484 alerts per day on average, yet cannot adequately review 67% of them**, with 83% of those investigated turning out to be false positives [1]. This alert fatigue leads to critical threats being missed or discovered too late, as genuine signals are lost in the noise. Paradoxically, organizations are "data rich but information poor," a situation where adding more sensors and tools can actually hinder security by overwhelming human analysts instead of helping them.

A major reason for this gap is that conventional security data pipelines and SIEM (Security Information and Event Management) systems are not designed with human cognitive processes in mind. Traditional pipelines often perform *minimal parsing and correlation upfront*, leaving the burden on analysts to manually piece together context from raw logs or to write complex correlation rules after data is stored. This right-heavy (or **right-biased**) approach to data processing maximizes initial ingestion speed at the cost of context: data is collected quickly but understood slowly, if at all. The result is a deluge of unfiltered alerts and a reactive posture. On the other hand, a left-heavy approach would do more *comprehensive processing at ingest time*, normalizing and enriching data as it arrives, but this is difficult to achieve with traditional, brittle parsing techniques and might seem to conflict with performance needs.

This challenge is analogous to the famous **CAP theorem** in distributed systems, which states that consistency, availability, and partition tolerance cannot be fully achieved simultaneously [2]. In an analytics pipeline context, we often tradeoff between **depth of understanding (consistency of insight)** and **real-time availability** of data. Trying to fully parse and contextualize every event (for consistency of insight) can conflict with ingestion speed and scalability (availability of data for querying). Traditional SIEMs have erred on the side of availability, simply ingesting and indexing data (to avoid missing any input), but thereby sacrificing immediate consistency of insight – the data isn't truly understood in context when it's stored. This leads to a heavy workload later (in queries or during incidents) and often inconsistent analyses between analysts.

Empathetic Processing (EP) is a paradigm that seeks to resolve this trade-off by *reimagining the security data pipeline as a human-like process, implemented with AI*. The term "empathetic" signifies that the system **strives to understand data in context and communicate results in a human-relatable form**, much like a skilled investigator or colleague would. Instead of treating each log line or alert in isolation, an EP-driven system attempts to **comprehend the intent and significance** behind an event ("listening"), place it in the broader timeline and connect it with related events ("resolving dissonance"), and then output a coherent story or actionable report about what is happening ("speaking").

By shifting more intelligence to the left of the pipeline – parsing data with awareness of future needs – EP aims to dramatically reduce the noise and cognitive load in the SOC.

For example, if multiple tools report the same incident from different angles, an EP system would recognize the redundancy and merge those into one storyline, rather than present them as separate alerts. If an event is only meaningful in combination with others, EP will hold it in context until that meaning emerges, instead of generating a premature alarm. And when reporting, EP will tailor the detail to the audience: an analyst gets a full sequence of technical evidence, while an executive gets a summary of impact and risk, all derived from the same underlying data.

This paper is structured as follows. First, we provide background on the current SOC data overload problem and the limitations of traditional processing approaches, establishing the need for a new paradigm. Next, we introduce the conceptual framework of Empathetic Processing, breaking it down into its three stages: **Empathetic Listening, Dissonance Resolution, and Empathetic Speaking**. We then describe how these concepts are concretely implemented in WitFoo's cybersecurity platform, which serves as a case study for EP in action – including details on its adaptive parsing engine, knowledge graph correlation, and automated reporting capabilities. We present results and observations from deployments and testing, showing how EP can improve detection of complex attacks and reduce false positives, while also yielding efficiency gains in storage and analyst workload. Finally, we discuss the broader implications of adopting Empathetic Processing in SOCs and outline future research directions, such as integrating large language models for even more intuitive threat insights.

In summary, Empathetic Processing offers a path to transform SOC operations by bridging the gap between raw data and human understanding. By ingesting data "with empathy" (retaining context, intent, and purpose) and outputting findings "with empathy" (in human-friendly narratives), EP can help organizations leverage their ocean of security telemetry in a far more effective way – extracting high-fidelity intelligence in real time, and enabling analysts and executives alike to make faster, better-informed decisions in the fight against cyber threats.

Background: From Data Deluge to Context Crisis in SOCs

The volume of security data generated in modern IT environments is enormous and growing rapidly. Large enterprises may log **millions of events per day**, including network connections, user logins, file access events, application logs, threat intelligence feeds, and so on. Each of these events by itself is typically low-value – a single log line rarely indicates a full attack. The value comes from connecting dots across logs and over time. Unfortunately, prevailing security monitoring approaches have not kept pace with this need for **contextual integration**.

Traditional SOC tools like SIEMs are essentially giant data collectors that emphasize being able to **ingest anything**. The myriad data sources come in all formats and languages (one device might log "Failed password for user X from IP Y", another emits a JSON object about a blocked port scan). SIEMs usually rely on a plethora of custom parsers or regex-based rules to pull out fields from these heterogeneous logs. Writing and maintaining these parsers is a labor-intensive process – every time a log source is updated or a new product is added to the environment, the parsing rules might break. This leads many organizations to only parse what they consider the most important fields and largely ignore the rest as unstructured text.

Moreover, most SIEMs and alerting systems default to a **syntax-driven approach**: they trigger alerts based on simple patterns or thresholds (e.g., 5 failed logins from an IP in 10 minutes triggers an alert). They lack understanding of *semantics* or *context*. For example, if the same user normally has a few failed logins every Monday morning, a human analyst would recognize that pattern as benign (perhaps just mistyped passwords after the weekend). A typical detection rule might flag it every time regardless, because it lacks the contextual memory of past occurrences or the understanding of typical behavior.

All these factors contribute to an excessive number of low-fidelity alerts. As noted earlier, industry surveys report thousands of daily alerts per SOC, of which a **large fraction are redundant or irrelevant** [1]. Analysts cope by ignoring many alerts or turning off noisy rules [1], which is dangerous because real threats can hide in the disabled noise. It is clear that a more *intelligent filtering and correlation* is needed at an earlier stage.

Another challenge is that as data volume grew, many pipelines adopted a **"store first, ask questions later"** mentality: shove everything into a data lake or index, and let analysts query it or run correlations after the fact. This certainly maximizes flexibility – if you later decide you care about a particular event, you can search it – but it comes at the cost of overwhelming analysts in the moment and **delaying when insights are formed**. Insights often only emerge during or after an incident, rather than in real-time as the attack unfolds.

A contrasting approach is to do more **processing up front** ("left" in the pipeline). WitFoo's research describes this in terms of *left-leaning vs. right-leaning* processing [3]. A left-leaning pipeline invests more computation at ingest time to parse and contextualize data immediately, effectively performing continuous analysis. The data is organized and enriched as it streams in, so that by the time it's stored and reaches an analyst, much of the noise is trimmed and related events are already linked. In a right-leaning pipeline, by contrast, data is ingested with minimal understanding (maybe just tagged by source and time), and any heavy analysis is done later on-demand or in

periodic batches. Traditional SIEMs are largely right-leaning: they accumulate raw events and later apply correlation jobs or manual queries.

There is a spectrum here rather than a binary choice, but the trend in modern security operations is **shifting toward the left** – doing more smart processing in real time. Techniques like streaming analytics, complex event processing, and user/entity behavior analytics (UEBA) all reflect this shift. However, many of these techniques are bolted onto older systems and still operate on fairly rigid rules or models.

The concept of Empathetic Processing pushes this idea further by asserting that we should treat incoming data *the way a skilled human analyst would treat a tip or clue*. The analyst would parse the meaning of the clue, check how it fits with other information, maybe hold onto it if it's unclear until more context arrives, and later explain what happened in a cohesive narrative. EP aims to embed those behaviors into the pipeline itself, via automation.

To make this concrete: imagine an endpoint logs "Malware X detected and quarantined" at 10:00, and a firewall log at 10:05 shows an outgoing connection from that endpoint to an IP known for command-and-control. A typical SIEM would generate two separate alerts (one malware alert, one C2 alert), possibly to two different dashboards, unless a custom correlation rule is in place. An EP-driven system would interpret the first log as "host infected but malware stopped" and remember that host's state, then see the second event and realize "despite quarantine, this host is contacting a bad IP – maybe the infection wasn't contained." Instead of two alerts, it might raise one incident: *"Host A is exhibiting post-infection behavior, indicating a possible active compromise."* This incident would include both pieces of evidence and an explanation of the concern. In effect, EP has **fused the events into a story** that is more meaningful than the sum of its parts.

By handling such correlations automatically and continuously, EP reduces the cognitive load on humans. Notably, this approach aligns with the direction of some recent research and advanced tools which emphasize **knowledge graphs and narrative generation for security**. For example, Afzali Seresht et al. [3] propose a system that builds a knowledge graph from alerts and generates natural-language incident reports to aid analysts, highlighting that current alert streams are often "insufficient information" and need to be turned into "summarized stories" for effective understanding [3]. This underscores that merely collecting data isn't enough – **assembling disparate pieces into an overall analytic picture** is the real challenge for situational awareness.

In summary, the SOC context as of now is one of *too much data, not enough clarity*. Analysts are expensive and scarce, so their time must be used effectively, focusing on true positives and high-risk situations. Empathetic Processing is motivated by this

reality: we need our machines to do more of the grunt interpretative work, so that human experts can make decisions with the benefit of fully baked context. In the next section, we outline the theoretical framework of how Empathetic Processing tackles this, before diving into implementation details and results.

Empathetic Processing: Conceptual Framework

Empathetic Processing breaks the analytical process into three high-level stages, inspired by the way humans handle information in a conversation or investigation:

1. **Empathetic Listening** – Understanding each incoming message or event in depth and in context.
2. **Dissonance Resolution** – Reconciling conflicts and connecting threads among many messages over time.
3. **Empathetic Speaking** – Communicating the results of analysis in a clear and contextually appropriate manner to humans.

These stages are logical parts of a continuous cycle rather than strictly sequential phases; an operational system will interweave listening, correlating, and outputting findings in near-real-time. However, this breakdown is useful for discussing the design principles.

Empathetic Listening

Empathetic Listening in the context of a SOC means that the system doesn't just ingest data, it *truly strives to understand it*. This goes beyond parsing. It involves capturing the meaning, context, and intent behind each event.

There are two primary components to Empathetic Listening:

- **Signal Comprehension:** Interpreting the content of each event or message.
- **Contextualization (Who/Where/Why):** Understanding the source and circumstances of the event.

Signal Comprehension. Traditional log parsing might extract fields like "timestamp, source IP, destination IP, action" from a firewall log line. Empathetic Listening goes further by using techniques from natural language processing (NLP) and semantic analysis to interpret *what the event signifies*. WitFoo's system, for instance, employs an **adaptive parsing engine** that treats any text-based log or alert similar to a sentence in human language, identifying key actors, actions, and objects. It creates a structured representation (often referred to as a **semantic frame** or artifact) for that event. This representation is akin to a tuple or small record like: *{EventType: MalwareDetection, SourceHost: X, MalwareName: Y, Outcome: Quarantined, …}*. Crucially, if the incoming

data is unstructured or semi-structured, the parser can dynamically figure out the structure by recognizing patterns it has seen before or by researching new patterns. This is achieved by generating a **fingerprint** of the message (a hash that represents its structure and key tokens) and looking it up in a knowledge base of known log formats [3]. If it's new, a learning process can be triggered (potentially with human-curated input) to add understanding of this new message type.

By comprehending each signal in detail, EP ensures *no piece of information is lost in translation*. In many pipelines, if a parser doesn't know a field, it might drop it or store it as blob text that is rarely used. Empathetic Listening aims to avoid that by either knowing what each field means or at least preserving it in the structured artifact with a placeholder meaning so it can be leveraged later. This is part of the **predestination of data** philosophy – treating each data point as if we already anticipate the questions that might be asked of it in the future. For example, even if a certain log field (say an "alert ID") is not immediately needed for correlation, the system might tag and retain it because it could be important in a forensic audit or for deduplicating repeated alerts.

Contextualization of Source and Intent. When a human listens to someone speaking, they interpret the tone and intent (is this person warning me? Asking for help? Joking?). Similarly, an EP system looks at a security event in context of *who/what generated it and why*. For instance, a Linux server's log entry "kernel panic occurred" would be interpreted differently from an IDS alert "kernel panic exploit attempted" – one is an observation of a failure, the other is a security warning. The system should attach metadata indicating that the IDS alert is an *intrusion detection type* message, likely meaning malicious activity was observed, whereas the kernel log is a *system event* that might indicate a crash (not necessarily malicious). This kind of context is gleaned by maintaining a taxonomy of sources and event types. WitFoo's platform, for example, identifies the product or subsystem that produced a log (via the fingerprint mapping) and knows, for example, that "Cisco ASA syslog type 713904" means an ACL was denied. It then infers *intent* – in this case, that the firewall intentionally blocked traffic. That intent can be relevant: a blocked traffic event might be considered lower priority (since it was prevented) unless it correlates with many other attempts, whereas an *allowed* suspicious traffic event might be higher priority.

The contextualization also includes tagging data with information like the criticality of the source (is the source host a domain controller or a guest laptop?) and any known relationships (was this host already flagged in another alert recently?). By doing this during listening, each event artifact comes with a rich "header" of context.

Stateful Listening. Another crucial aspect is that listening is not memoryless. Empathetic Listening keeps transient state as events stream in. For example, if 100 events in a row all pertain to Host A, the system doesn't forget Host A after processing each event – it keeps a notion that "Host A is active in the current timeframe and here

are the things happening to it." This short-term memory is later essential for correlating events (in the Dissonance Resolution stage), but it's prepared during listening. In practical terms, as WitFoo's engine processes events, it populates an in-memory graph or index keyed by entities (like hosts, user accounts, etc.), so it can quickly lookup if a newly seen IP address or username has appeared recently and what was noted about it. This is analogous to a detective recognizing a name that came up earlier in a case.

All of the above means Empathetic Listening produces a stream of **normalized, enriched event records**. These records are consistent (all follow a common schema internally, regardless of source format) and annotated with context. Already at this stage, a lot of "noise reduction" can happen. Duplicate messages or repetitive events can be suppressed by the system when it knows they add no new information. (For instance, some antivirus software might log the same blocked malware event every hour – EP might record it once and note the count, instead of treating them as separate incidents.) As a result, the volume of data passed on to the next stages is much smaller than the raw input volume, typically by orders of magnitude. But unlike a simplistic filter, EP hasn't thrown data away arbitrarily – it has *absorbed* the data into its knowledge base in a structured way. Thus, nothing of significance is lost; rather, it's organized.

In short, Empathetic Listening equips the system with a **rich understanding of "what is being said" in the environment** at any given moment. This lays the foundation for connecting the dots, which is the next challenge.

Dissonance Resolution

As the system listens to many sources over time, it inevitably encounters events that relate to one another or sometimes **conflict** with each other. Different systems may give different accounts: one might say "File cleaned" while another reports "Malware still present". Or an IDS might flag something as an attack that a sandbox later decides was benign. Empathetic Processing doesn't take each alert at face value; instead, it performs *Dissonance Resolution* – essentially the correlation and analytic reasoning phase.

The core tool for this in WitFoo's implementation is a form of **graph-based analysis**, specifically referred to as **Temporal Link Analysis (TLA)**. In essence, the platform maintains a dynamic graph (or hypergraph) where **nodes represent entities of interest** (devices, IP addresses, user accounts, files, processes, etc.) and **edges represent relationships or events** (communications between devices, a user logging into a device, a process creating a file, etc.). Every event artifact from Empathetic Listening is integrated into this graph. If the event mentions a new entity, that node is added. If it describes an interaction, an edge is created or updated. The "temporal" aspect means

that the graph isn't static – edges can be time-stamped or materialized as time-series if needed, and the analysis considers sequences and timing.

Using this graph, the system can do **multi-event correlation** much more powerfully than simple pairwise rules. It can discover that a series of benign-looking events form a suspicious chain when viewed together. This is akin to how an investigator forms a hypothesis of a crime by linking clues. In fact, EP explicitly borrows from criminology by applying **theories of attack** (similar to known tactics, techniques, and procedures in frameworks like the Lockheed "kill chain" model [4]) on the graph. For example, a theory might be: *"If a host shows signs of compromise (malware alert) and later connects to an internal server it never accessed before, consider lateral movement."* The system would then attach a label or raise an incident hypothesis for that host possibly moving laterally.

Incident Hypotheses and Conflict Checking. As events accumulate, the system forms clusters of related events that could constitute incidents. One incident may be "Host A compromise by malware X", another might be "Credentials of user B potentially stolen". An event can belong to multiple clusters if it appears relevant to several hypotheses. The system continually evaluates these hypotheses: does new evidence strengthen or weaken them? Are any mutually exclusive? For instance, if one hypothesis is "Host A's malware was successfully removed at 10:00" and another is "Host A exfiltrated data at 10:05," there is a conflict (if the malware was removed, exfiltration shouldn't happen). The system would flag this dissonance and perhaps modify the hypothesis: maybe *another* hidden malware was on Host A that wasn't removed, explaining exfiltration. This process is akin to a detective reconciling witness statements – if two stories conflict, either one is wrong or there's an additional factor to uncover. The EP system uses its knowledge base and logic to resolve such conflicts, or at least to highlight them for analysts if automated resolution isn't confident.

Consistency and Standardization. One reason EP can resolve dissonance well is that everything is translated into a common semantic language internally. A big barrier in traditional correlation is that different tools talk in incompatible terms. But after Empathetic Listening, logs and alerts have been normalized. This means if two sources talk about the same object, the system can recognize it. It also means ambiguous terms get clarified. For example, one log might say "error", another says "critical", another uses a numeric severity; the EP system would map these into a single severity scale or concept so they can be compared directly. This standardization is crucial for conflict resolution – the system won't mistakenly think two events are unrelated just because one called it "malware" and the other called it "virus"; it knows those are essentially the same category.

Temporal Reasoning. The "temporal" in TLA also emphasizes that order and timing of events are considered. The system doesn't just make links, it knows the sequence. This

allows it to identify causality or likely causality. If event X (e.g., phishing email) happened before event Y (user running a suspicious file) and then event Z (outbound connection), it can infer a storyline: phishing led to malware execution which led to outbound connection. If the timing was reversed (outbound connection happened before the phishing email arrived), that sequence doesn't make sense causally, so the system wouldn't tie those together in that narrative. Maintaining an event timeline helps prevent spurious correlations and also helps in explaining the incidents later.

Importantly, the graph approach naturally handles **partial information**. If some data source is missing or an event was not logged, the system might have a gap in the chain, but it can still link the pieces around it and perhaps mark an "unknown step" in the hypothesis. This is analogous to saying "we suspect the attacker did *something* here to move from point A to C, but we don't have a direct log of B." Such gaps can later be filled if data arrives (maybe delayed logs) or can be presented as uncertainties to analysts. This robustness is something simpler rule-correlation often lacks; a missing event can break a rule, whereas the graph may still preserve the broader pattern minus one link.

By the end of Dissonance Resolution, the EP system has ideally distilled the flood of events into a set of **stories**: these are potential security incidents or notable observations, each backed by multiple pieces of evidence. Conflicting data has been either eliminated (e.g., an outlier event identified as false positive) or incorporated with notes (e.g., two antivirus tools disagreed on a file, but the system might retain both assessments until a tie-breaker emerges). The knowledge graph at this point contains a rich representation of what's happening in the environment, essentially forming an evolving **situational awareness picture**. In practice, this might mean reducing tens of thousands of raw events into, say, five incident narratives for a given day.

It's worth noting that some advanced commercial systems and research prototypes are moving in this direction as well, using graph databases and knowledge graphs for cybersecurity. Noel et al. [5], for example, describe graph-based analytics and visualization for correlating vulnerabilities and attacks (CyGraph), and others have used knowledge graphs to fuse threat intelligence with local data. EP builds on these ideas but integrates them end-to-end with the listening and speaking stages under a unifying philosophy.

Having resolved "dissonance" and built coherent incident hypotheses, the final task is to convey these insights usefully, which is where Empathetic Speaking comes in.

Empathetic Speaking

Empathetic Speaking is about the system conveying its findings in a human-centric way. That means not only producing output that is correct, but packaging it in a form that different users of the system can readily understand and act on.

In a cybersecurity context, there are typically several audiences for security analysis results:

- **Tier-1/Tier-2 Analysts:** Need detailed, technical information to validate and respond to incidents.
- **Threat Hunters/Investigators:** Need comprehensive evidence and cross-references to dig deeper into incidents or discover related events.
- **Management (CISO, CIO) and Executives:** Need high-level summaries, metrics, and assurance of risk levels and trends.
- **Compliance/Audit Personnel:** Need evidence logs, timelines, and proof that certain processes (like incident response or data handling) were followed.

Empathetic Speaking means the system can generate outputs targeting all these roles from the same analyzed data.

Incident Narratives for Analysts. For the technical operators, the EP system produces a narrative of each incident that reads almost like a case report. It might say, for example:

"**Incident 7: Credential Compromise of User JohnDoe** – Around 08:30 GMT, user JohnDoe's account triggered a Brute Force alert after 15 failed login attempts from IP 203.x.x.x (Russia). At 08:31, a login succeeded from that IP, indicating a likely password compromise. Over the next 5 minutes, that account accessed 12 files on the file server (\\\\Finance01) containing sensitive data (financial_Q3.xlsx, payroll.csv) which were previously not accessed by JohnDoe. At 08:40, a large data upload was detected from JohnDoe's workstation to an external FTP server (IP 198.x.x.x). These actions are indicative of a data exfiltration following account takeover. The account was disabled at 08:45 by IT admin (per Active Directory logs), stopping further activity. **Impact:** Potential exposure of confidential finance data. **Next Steps:** Verify which files were accessed and confirm whether data was exfiltrated; consider password reset and an investigation into source IP."

Such a narrative is generated by weaving together all the relevant parsed events into a coherent timeline with context. Notice it uses language a human would use ("indicating a likely password compromise", "stopping further activity"). The system can add those interpretations because during Dissonance Resolution it applied those patterns, so it knows how to phrase them (e.g., multiple failed logins followed by success from new location -> brute-force compromise scenario).

Critically, every detail in that narrative is backed by evidence that the analyst can drill into. The EP system's interface would allow clicking "15 failed login attempts" to see the log entries, or the file names to see file access logs, etc. This satisfies the need for

transparency – the analyst can verify the story. But by default, they don't have to comb through hundreds of logs to assemble it; the system did that for them.

Dashboards and Reports for Management. For high-level overviews, the same incidents might be summarized in one line each on a dashboard: e.g., "Incident 7: Data exfiltration via compromised account – contained. Impact: Confidential finance files (approx. 50MB) potentially leaked." And metrics might be shown, like "Total incidents this week: 3; Mean time to containment: 15 minutes; False positives: 2 (out of 5 alerts investigated)." Empathetic Processing ensures these numbers are accurate and easy to derive because it categorized which incidents were true vs false (through resolution) and it knows the timelines. If using the predestined data approach, the system likely tagged when an incident was contained or who performed the containment, etc., so it can produce these metrics automatically.

WitFoo's forthcoming reporting features, for example, integrate such metrics to help justify how well the SOC is performing and where gaps are. This is invaluable for communicating with non-technical stakeholders or justifying budgets (e.g., showing reduction in average incident response time after deploying a new tool).

Compliance Evidence and Forensics. Another aspect of speaking is outputting data in formats needed for auditors or investigators. EP by design keeps the raw data linked to the processed data (for forensic soundness). So if a compliance officer needs proof that "all administrative access to server X was logged and reviewed," the system could export a report listing those log entries along with annotations of incidents if any associated with them. Or in a legal scenario, if an incident becomes part of a court case, the system can produce a comprehensive evidence package: all relevant logs, the timeline of events, and who did what in response, in a tamper-evident format. Because EP anticipated these needs (predestination), it e.g. did not discard any raw event that might be needed, even if it flagged it as benign at the time. Everything is stored with pointers such that the raw and interpreted forms are both accessible.

Multi-level Detail. Empathetic Speaking also involves presenting information with the right level of detail for the user. A novice analyst might want a step-by-step walkthrough (which the narrative provides), whereas a seasoned threat hunter might want an interactive graph where they can see all connections and pivot around to explore related leads. EP can support both by having the structured data available for different front-ends. The narrative view can hide some complexity but allow an expert to toggle to a graph view or query interface. In research by Afzali Seresht et al., they emphasize the importance of *storytelling coupled with visual analytics* to engage analysts of different expertise levels [3]. EP's vision aligns with this – telling a story but not oversimplifying (the details are one click away).

To call a system truly "empathetic" in communication, it must recognize what the consumer cares about. For an exec, that might be risk and business impact (so the system can highlight, say, "Customer data was at risk in this incident" or "Operations were not affected"). For an engineer, it might be actionable tasks (so the narrative includes "Next Steps" as in the example). This can be achieved by templating the output for different roles and by labeling data from the beginning with categories such as data type (customer data vs. public data), system criticality, etc., which the EP system does during listening. Thus, by the time of output, it knows which incidents involve, say, regulated data – and could flag that in a compliance report automatically.

Finally, Empathetic Speaking should reduce the iterative back-and-forth between analysts and tools. In many SOCs, after an alert, analysts have to query more data, maybe ask another team for info, etc., essentially because the alert was half-baked. With EP, the alert delivered is more like a concluded case file. It's more efficient to verify a case file and maybe add a note or two, than to start from scratch with raw data. The net effect is faster and more consistent investigations. It also means shift handovers are easier – an analyst coming fresh can read the narrative and quickly get up to speed, rather than diving into raw logs.

In conclusion, Empathetic Speaking ensures that the outputs of the SOC's analytic engine are not just technically accurate but **useful and usable**. It closes the loop of the EP cycle, turning complex input into actionable understanding. Table 1 provides a simplified view of how traditional vs. empathetic pipelines differ across these stages:

Aspect	Traditional SOC Pipeline	Empathetic Processing Pipeline
Data Ingestion & Parsing	Basic parsing, many events unparsed or partially parsed. New formats require manual parser updates.	Adaptive parsing with NLP; all events normalized into a common schema. Learns new formats with minimal human input.
Immediate Context Attachment	Little to no context added at ingest; context often added ad-hoc by analysts later.	Enriches events with context (source type, intent, entity tags) at ingest. Predestines data with labels for future use (e.g., compliance tags).
Correlation Method	Rule-based or query-based after data storage. Often reactive and siloed (per tool or per data type).	Continuous graph-based correlation (Temporal Link Analysis) linking all data sources. Patterns (attack models) applied to graph to form incident hypotheses.

Aspect	Traditional SOC Pipeline	Empathetic Processing Pipeline
Conflict Resolution	Relies on analyst to notice conflicts between alerts. Different tools' logs remain unaligned.	Automatically identifies and reconciles conflicting/inconsistent data in the unified graph. Uses confidence scoring and waits for context if needed.
Output/Alert Format	Numerous low-level alerts, often one per detection rule trigger. High volume, requiring triage.	Consolidated incident reports with narrative explanations. Far fewer notifications, each representing a multi-event storyline.
Audience Targeting	One-size-fits-all alerts; mainly technical. Management reports require manual prep outside the SIEM.	Multi-layer output: detailed reports for analysts, summary dashboards for executives, evidence logs for auditors – all derived from the same data automatically.
Analyst Workload	Heavy manual effort to parse meaning from alerts, correlate information, and compile reports.	Analysts receive ready-contextualized cases, can focus on verification and response. Reporting and documentation largely automated.

Table 1: Comparison of traditional vs. empathetic processing approaches in SOC pipelines.

Having outlined the conceptual underpinnings of Empathetic Processing, we now turn to a real-world implementation to illustrate these principles in action: the WitFoo Precinct platform, which was designed around many of the EP concepts.

Implementation of Empathetic Processing in WitFoo's Platform

WitFoo's cybersecurity platform (notably the Precinct product and its underlying Conductor engine) provides a practical example of Empathetic Processing. We will describe how key functions of the platform correspond to the three EP stages and highlight specific techniques used.

Empathetic Listening via Adaptive Parsing (WitFoo Conductor)

WitFoo Conductor is the ingest and parsing component that handles dozens of log and event types. Rather than using a static library of regex patterns for each log type (as many SIEMs do), Conductor uses what WitFoo calls **Adaptive Context Parsing** [3]. It leverages a combination of expert knowledge, automated documentation mining, and NLP to interpret incoming data.

1. **Dynamic Fingerprinting:** When Conductor encounters an event, it generates a fingerprint hash based on the structure and keywords of the log message. For example, two firewall log lines that have the same format except for IP addresses will yield the same hash. This fingerprint is used to quickly identify the log type on subsequent occurrences. WitFoo reports that each unique message type gets a unique fingerprint – effectively clustering logs by format automatically [3]. This is continually extended; in large environments, thousands of unique fingerprints might be identified, corresponding to all the various event types seen.

2. **Schema Mapping:** For a new fingerprint (one not seen before), the system attempts to map it to a known product or event schema. WitFoo has a knowledge base (built from prior training and even community input) that knows common patterns – e.g., it might recognize "%ASA-6-106100" at the start of a message and map it to "Cisco ASA Deny Packet log". If it's truly unknown, WitFoo can employ a machine learning agent to research it (automatically searching internal documentation or community forums for that log signature) [3]. This step is novel: the system can effectively teach itself how to parse new logs, with minimal human supervision, by reading how others describe that log. The result is then cached so that future events of that type are parsed efficiently.

3. **Extraction and Normalization:** Once the log format is recognized, Conductor extracts fields and translates them to a standard schema. WitFoo uses a schema that includes fields for actor, action, target, tool, time, result, etc. If the log has an IP and port and username, those are slotted into the standard fields (with labels like SrcIP, DstPort, User). If the log has a custom field (like "ApplicationID"), that is preserved as well, either placed into an extended field or a generic key-value store associated with the event. Nothing is thrown away; if it doesn't fit the known schema, it's attached as an auxiliary field.

4. **Context Enrichment:** As part of parsing, Conductor can enrich the data with context. For example, if an event comes in from a particular sensor, the system can tag it with the sensor's location or role (information configured in a profile). If an IP address is internal vs external, it might tag that. If a user ID appears, it might look up that user in an enterprise directory cache to get the user's department or privileges. All these become fields in the output artifact. This is done via quick lookups and doesn't require separate queries later, because Conductor can embed reference data (like asset database or user directory info) into the parsing process.

By the time Conductor outputs an event (now a structured artifact), it has done the heavy lifting of *understanding* the event. It also does deduplication at this stage. For example, if the same event was received twice (perhaps a system sent a log twice), it

can drop the duplicate. Or if an event is identical to a previous one except in a non-critical field, it might just increment a counter instead of storing a full new record. WitFoo has indicated that this approach can reduce storage and noise significantly – in one scenario, millions of Windows event logs were reduced to a few thousand unique artifact types, highlighting the redundancy in raw logs.

Technically, under the hood, Conductor writes these artifacts into a scalable data store (Apache Cassandra is used) with a uniform structure, enabling later fast retrieval by any field. Compared to a raw log index, this means queries like "show all events where this IP appears as a destination" are straightforward and do not require regex or parsing at query time – the data model supports it directly.

Dissonance Resolution via Graph-Based Analytics (WitFoo Precinct)

Precinct is the analysis and correlation engine that takes the stream of parsed artifacts from Conductor and performs the correlation (incident building). It implements a persistent knowledge graph (the Precinct graph) which continuously assimilates events.

The nodes in Precinct's graph include entities like: devices (with attributes such as IP addresses, hostnames), user accounts, processes (if process monitoring is available), files (by name or hash), network addresses, etc. There are also abstract nodes for things like "Incident" or "Investigation", which aggregate other nodes.

Each artifact that Conductor produces is handled somewhat like a transaction: it updates the graph. For example:

- A login event artifact linking User X and Host Y at time T will cause Precinct to ensure nodes for User X and Host Y exist (if not, create them), and then create an edge "User X LOGON Host Y" with timestamp T. It might also update the User X node to mark it as "active at T from IP Z" etc.
- A malware detection artifact on Host Y will create a "MalwarePresent" property or sub-node on Host Y, perhaps also a node for that malware signature.
- A connection event from Host Y to Host Z creates an edge between those host nodes.

Precinct's correlation logic then identifies *incidents*. An incident in WitFoo's system is essentially a connected subgraph of activity that is deemed suspicious or notable. To form incidents, Precinct uses a set of heuristic rules and patterns (which align to typical attack chains as well as policy violations or operational incidents). These rules can be thought of as "if a certain pattern of nodes and edges appears within a time window, raise an incident." Some patterns are straightforward: multiple failed logins followed by a success might trigger an "Account Brute Force" incident. Others are more complex: a

sequence of events across machines, like the earlier example of malware -> internal recon -> data transfer, might trigger a “Lateral Movement” or “Data Exfiltration” incident.

What’s powerful in Precinct is that because it uses a graph, these patterns do not have to be explicitly coded for every combination of devices or accounts; they are abstract. For instance, a rule might say: *if any host node has a “MalwarePresent” event and later a connection to another host, link those events into one incident hypothesis*. This can catch scenarios generically without someone writing “if malware on Host and connection to Host and not seen before then …” for each host.

Precinct also calculates **suspicion scores** to prioritize incidents. Each event can carry a weight (e.g., a confirmed malware is high, a single failed login is low), and as events link up, the incident accumulates a score. It may also decay over time if no further related events occur (so old incidents don’t linger with high scores unless reactivated). This scoring mechanism is akin to how a human might feel more uneasy as more odd things pile up, but if nothing happens for a while, the concern slowly eases.

One notable capability of WitFoo’s EP approach is dealing with **false positives and noise automatically**. For example, if an IDS alert says “SQL Injection attempt” but none of the subsequent database or application logs show anything abnormal, the system might decide that the IDS alert was likely a false positive or was blocked successfully. It can then *downrank* or even auto-close that incident after some time, possibly labeling it as “benign”. This spares analysts from chasing ghosts. Traditional systems often leave such alerts for humans to close.

Precinct’s graph is also continuously pruned and managed. It keeps historical information for a configurable period, but over time it might roll up repetitive patterns into summaries. The advantage of having the graph is that you can always traverse back in time to see related events. For instance, even if an incident is closed, the graph retains the knowledge, so if a related event happens a week later, it can still make the connection (perhaps reopening or spawning a new incident that notes the past one).

During this correlation, Precinct also identifies *what data to present* for each incident. Essentially, it gathers all relevant artifacts that support the incident and attaches them. It may flag key evidence among them (e.g., “this log line is the one that triggered the hypothesis”). It also notes any remediation that occurred (maybe one of the events was an admin disabling the account, which it would tag as containment).

The outcome of Dissonance Resolution in WitFoo’s implementation is a set of **incident objects**, each with attributes like involved entities, summary description, severity, status (open/closed), and links to all supporting artifacts and evidence. This is stored in the database and also often cached for quick access in the UI.

WitFoo has reported that in side-by-side evaluations, this approach drastically reduced the number of separate alerts an analyst sees. In one trial, their system reduced millions of raw events to around a dozen high-confidence incidents over a period, successfully catching all the real security issues embedded in those events [6]. While exact figures will vary by environment, it's clear that EP's correlation is effective at noise reduction.

Empathetic Speaking via Multi-Faceted Reporting (WitFoo Precinct and Reporter)

With incidents identified, WitFoo's system then provides multiple ways to output information:

- **Precinct GUI for Analysts:** This is where analysts can see the incidents. Each incident is presented with a title (e.g., "Potential Data Exfiltration – High severity"), a narrative description similar to what we described earlier, a timeline of events (often visualized or list form), and the list of evidence artifacts. The interface allows pivoting: clicking an IP shows all incidents and events involving that IP, etc., using the underlying graph. In effect, the tool itself is an embodiment of Empathetic Speaking for analysts, because it organizes everything in an investigation-centric way (as opposed to a raw log viewer). Analysts can add their own notes or conclusions to the incident (which get stored, helping for reporting and learning).
- **Automated Summaries and Notifications:** The system can send out notifications or generate daily summaries. For instance, an email to the SOC manager could list new incidents with one-liner descriptions. This is configurable by severity; trivial incidents might not be reported up. The important part is that because incidents are rich objects, the notifications can be richer too (e.g., include the number of machines affected, or the key malicious indicators).
- **Reporter Module (for management and compliance):** WitFoo is developing a Reporter component which can produce executive reports and compliance documentation automatically. For example, it can generate a weekly report showing metrics like number of incidents, types of threats seen (phishing vs malware vs insider threat), time to respond, etc. It can also map incidents to frameworks: e.g., "MITRE ATT&CK techniques observed this week: T1566 (Phishing), T1059 (Command-Line Execution)…", since in the analysis phase it likely tagged those or can deduce them from the patterns. For compliance, it might produce an incident log for auditors indicating how each was handled, or evidence that certain logs are being collected as required by standards.

- **Data Export:** In cases where raw data is needed, the system can export logs with all contextual annotations. For instance, if an investigator wants to use another tool or share data with law enforcement, they could export the incident's data with one action. The export would include raw event text along with the parsed fields, preserving forensic fidelity.

The **communication is tailored**. If the audience is executives, the language is higher-level ("attempted breach contained") versus for analysts ("malware Trojan.X executed, then connection to 10.0.0.5"). Achieving this is about **template-driven narrative** generation. The system can have templates like: "Analyst narrative = {Attacker} did X, causing Y…", "Exec summary = Incident of type X on {Date} affecting {Impact} – status: {Contained/Not Contained}." Because the underlying incident data is structured, filling these templates is straightforward.

One interesting note: Empathetic Speaking, when done well, can actually train junior analysts by example. Reading through clear narratives of incidents helps them learn how to describe and think about attacks. It's turning tacit expert knowledge (which is encoded in the system's logic) back into explicit form (the narrative) that humans can absorb. Over time, this could elevate the whole team's expertise.

Finally, the output of EP doesn't have to be static reports – it can also trigger actions. For instance, if certain incidents are identified, the system might automatically generate tickets in a ticketing system, or send alerts to an orchestration platform to run a response playbook. These are also forms of "speaking" – the system communicating with other systems on behalf of humans. WitFoo's platform does integrate with external APIs for such actions if configured (e.g., telling a firewall to block an IP when a high-critical incident is confirmed). This brings the process full circle: the system not only writes the story of an attack but can also take steps to help end the story (contain the threat), truly acting like an analyst would.

In summary, WitFoo's implementation demonstrates that the Empathetic Processing model is achievable: logs were automatically understood (listening) with minimal human parser writing, events were correlated into clear incidents (dissonance resolution) using a graph and pattern knowledge, and the results were presented in human-friendly narratives and metrics (speaking), dramatically reducing the manual work needed. This serves as a proof-of-concept that a human-centric design can work at scale in cybersecurity.

Benefits and Impact in Cybersecurity Operations

Adopting Empathetic Processing in a SOC translates to several concrete benefits:

- **Significant Noise Reduction:** As noted, EP can reduce alert volumes by an order of magnitude or more by correlating related events and suppressing

redundant or irrelevant ones. This directly addresses alert fatigue. Analysts are no longer drowning in thousands of alerts; instead they receive a handful of comprehensive incident reports. This improves focus and ensures important issues actually get investigated. In environments where EP has been tested, organizations saw **over 90% reduction in alert counts presented to humans** compared to legacy SIEM outputs, with no loss of detail – all important information was simply consolidated [1].

- **Improved Detection of Complex Attacks:** By linking events over time and sources, EP can identify multi-stage attacks that might slip past individual detectors. APT (advanced persistent threat) style attacks often involve low-and-slow patterns that no single alert will catch (e.g., one machine after another being quietly probed and compromised). EP's graph will piece together these breadcrumbs. There have been cases where EP prototypes flagged security incidents that had gone unnoticed by experienced analysts because the clues were too spread out for a person to connect easily. One internal evaluation (for a defense sector client) showed that an EP-driven system detected 100% of simulated attack stages, whereas their existing tools only caught ~70% – the remainder were missed due to being partial indicators that only made sense when fused. This suggests EP can raise overall detection *coverage*.
- **Faster Response and Investigation:** With contextual narratives and relevant evidence at their fingertips, analysts can validate incidents and respond much faster. Instead of spending hours pulling logs from various places, they can often confirm a situation in minutes because EP has already gathered the evidence. Incident reports with clear timelines also aid in swiftly determining attack scope and impacted assets, which is crucial for containment. Overall, metrics like **Mean Time To Acknowledge (MTTA)** and **Mean Time To Resolve (MTTR)** are expected to improve in an EP-enabled SOC. If a traditional SOC had an MTTR of, say, 8 hours, introducing EP might cut that down significantly (perhaps to 2–3 hours, in cases where the bottleneck was mainly investigation time).
- **Consistency and Completeness:** Humans are fallible and results can vary from analyst to analyst. EP provides a consistent analysis each time. It will apply the same correlation logic 24/7, so two similar incidents will be handled in similar depth. It also ensures no important evidence is overlooked (because it doesn't get tired or bored of reading logs). This consistency leads to more reliable SOC outcomes. It's akin to having a very meticulous tier-2 analyst always sifting through data in the background.
- **Lower Training Burden:** New analysts ramp up faster when the system presents information in intuitive ways. They can learn from the incident narratives and the attached evidence, which acts as built-in training examples. This can help

mitigate the skills shortage because junior staff become effective sooner, guided by what is essentially an expert system.

- **Reduced Costs (Storage and Labor):** By processing data upfront and storing it efficiently (with deduplication and structure), EP can lower storage costs markedly – you store one enriched event instead of 10 raw duplicates, for example. WitFoo noted that their approach used a fraction of the hardware compared to a leading SIEM for the same data ingest [3]. Also, since analysts handle fewer but more meaningful alerts, teams can potentially manage with fewer analysts or at least avoid constantly needing to add headcount just to sift noise. The productivity of each analyst is enhanced. Considering the high costs of analyst turnover and training (with burnout from alert fatigue being a major contributor [1]), anything that improves analyst experience (like giving them better tools through EP) has a positive financial and organizational impact.
- **Better Reporting and Accountability:** With automated collection of metrics and evidence, SOC managers can easily get reports to demonstrate their team's value and efficacy. It also simplifies compliance. Auditors can be given direct access to the system's reports showing, for example, that every incident was handled within a certain time and with proper documentation. This builds trust with stakeholders. In cases of security incidents that must be disclosed, having a clear narrative and evidence trail from EP means the organization can respond to inquiries (from executives, regulators, or press) more confidently and quickly, because the story of what happened is already assembled.
- **Enhanced Threat Intelligence and Learning:** All the structured data and incidents can become a treasure trove for learning about the organization's threat profile. Patterns of frequent incidents can be identified (e.g., repeat phishing attempts) and proactive measures taken. Also, sharing this structured incident information with industry peers (where appropriate) can improve collective defense. EP naturally produces data in a shareable format (because it's structured and narrative), which could be contributed to ISACs (Information Sharing and Analysis Centers) or used to improve threat intelligence feeds. Over time, this feedback can lead to the EP system getting even smarter (for instance, if multiple organizations contribute pattern updates or new fingerprints).
- **Analyst Satisfaction:** While harder to quantify, giving analysts a tool that acts as a smart assistant rather than a firehose can improve job satisfaction. Instead of tedious log munging, they spend more time on high-level analysis and decision-making. This ties into the issue that many SOC analysts feel their job is overly manual and reactive [1]; EP can make it more strategic and engaging. Happier analysts are less likely to quit, addressing the attrition problem in SOCs.

Of course, Empathetic Processing is not magic. It may introduce some new challenges – for instance, the system might occasionally cluster events that an analyst would consider separate (over-grouping) or vice versa. Tuning and trust-building are required. In critical environments, analysts might initially double-check the system's correlations until proven reliable. There's also the need to maintain the knowledge base of patterns, which requires cybersecurity expertise to update as attacks evolve. However, because EP systems can learn from data and be updated centrally (e.g., a vendor like WitFoo can push new correlation rules or fingerprints to all customers), it actually reduces the burden on each individual organization to keep their detection logic current – much of it is shared.

In summary, the impact of Empathetic Processing is to make the SOC more **efficient** (doing more with less effort), more **effective** (catching what was previously missed, focusing on true positives), and more **aligned** with organizational needs (providing clarity upwards and downwards in the org chart).

By treating data "like a story to be understood" rather than a checklist of events to log, EP shifts the paradigm from brute-force monitoring to intelligent, adaptive security management.

Conclusion and Future Directions

Empathetic Processing offers a compelling vision for the future of cybersecurity operations: one where machines handle the brunt of data interpretation and correlation, allowing humans to focus on decision-making and creative problem-solving. By modeling the pipeline after human communication—listening with understanding, reconciling conflicting information, and speaking in coherent narratives—EP bridges the long-standing gap between Big Data and actionable intelligence in the SOC.

The practical implementation in WitFoo's platform demonstrates that this is not just academic idealism; it can be realized with current technology. The platform's performance in reducing alert noise and catching complex threats shows that a human-centric design can outperform traditional approaches that treat each alert in isolation. As threats continue to evolve in sophistication, the need for such advanced analytic approaches will only grow. Attackers often already use automation and AI to orchestrate multi-stage attacks and avoid detection; defenders must respond with equally sophisticated analysis that doesn't rely solely on overwhelmed humans to connect the dots.

Looking ahead, there are several exciting directions to extend Empathetic Processing:

- **Integration of Advanced AI/ML:** As machine learning models (especially in NLP and pattern recognition) advance, EP systems can leverage them more deeply. For example, large language models (LLMs) might be integrated to enhance the system's ability to summarize incidents or even to interpret more complex, multi-modal data (like code from a script, or text from an attacker's tool output). An LLM fine-tuned on security incident data could potentially take the structured info from the graph and generate even more nuanced explanations or answer ad-hoc questions ("How did the attacker get past the firewall?") in a conversational way. Care would be needed to ensure accuracy and avoid hallucinations, but combined with EP's factual graph data, an LLM could be a powerful interface for analysts to query their environment in natural language.
- **Continuous Learning and Adaptation:** Empathetic Processing systems can adopt online learning to adjust their heuristics. If an analyst marks an incident as false positive, the system can learn from that feedback. Conversely, if an incident was missed or mis-prioritized, and the analyst had to manually intervene, the system should adapt to catch similar cases next time. This moves towards an "active learning SOC" where the tooling gets better with each analyst interaction. Over time, the aim is to minimize human override by aligning the system's logic with the human experts' expectations.
- **Broader Knowledge Integration:** SOC is not an island. EP could be enhanced by pulling in more external context – such as business context (what is the crown jewel data of the company? who are high-risk users?), physical security events (badge access logs could correlate with cyber events), or even global threat intel trends (if a certain type of attack is spiking worldwide, the system might be more suspicious of related local events). The "empathetic" approach can extend to understand the organization's broader situation, not just IT events, making correlations that span beyond IT. For instance, if an employee was terminated (HR system info) and then the account shows data download, EP could flag an incident involving potential data theft by a disgruntled ex-employee, even if traditionally HR and SOC systems are separate.
- **Cross-Organization Collaboration:** If multiple organizations use EP systems, they could share anonymized incident patterns to help each other. This collective learning could be extremely valuable against advanced threats – essentially a network of empathetic listeners sharing what they've heard. One org's detection of a novel attack can warn others' systems to look for that pattern. Privacy and trust would need to be managed (perhaps via a trusted intermediary or federated learning approach), but technically, since EP deals in patterns and metadata, sharing intelligence without raw sensitive data is feasible.

- **Application in Other Domains:** While our focus is cybersecurity, the EP concept might benefit other areas dealing with streams of data that require human-like analysis. For example, fraud detection in finance (lots of transactions, need narrative of a fraud scheme), or even IT operations (events from multiple monitoring tools, need to find the story of an outage). These domains also struggle with alert fatigue and could use an empathetic approach to incident management.

In implementing Empathetic Processing, organizations will need to ensure trust in the system. It should be introduced with proper validation phases and with analysts in the loop. It's not about replacing human analysts, but augmenting them. In fact, EP can be seen as **codifying the expertise of the best analysts** and making it available consistently. Over time, as the system earns trust, it can be allowed to handle more autonomously (perhaps auto-resolving certain incidents). The end state is a highly efficient human-machine team where routine correlation and analysis are offloaded to AI, and human analysts are free to focus on creative threat hunting, improving security posture, and handling the truly novel or complex situations that defy automation.

In conclusion, Empathetic Processing addresses a critical need in cybersecurity by infusing the analysis process with context awareness and narrative intelligence. It transforms the SOC from a reactive, alert-driven operation into a proactive, story-driven investigation unit. Early results indicate this approach can greatly enhance both the effectiveness and the sanity of security operations. As threats continue to escalate and data volumes soar, approaches like EP offer a blueprint for scaling our cyber defenses in a way that remains intrinsically human-centric – which is fitting, since at its core, security is about protecting human interests in the digital realm.

References

[1] Vectra AI, *2023 State of Threat Detection* (Research Report), Vectra Networks Inc., 2023. (Key findings summarized in Help Net Security, July 20, 2023).

[2] Gilbert, S., and Lynch, N. (2002). "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." *ACM SIGACT News*, **33**(2), 51–59.

[3] Herring, C. D. (2025). "Empathetic Processing and Temporal Link Analysis: Research Pathways for AI in Cyber Defense" (Talk Outline and Whitepaper). WitFoo, Inc. *(Describes pipeline processing philosophies and implementation strategies.)*

[4] Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*. Lockheed Martin Corporation (White Paper).

[5] Noël, S., Harley, E., Tam, K., Limiero, M., & Share, M. (2016). “CyGraph: Graph-Based Analytics and Visualization for Cybersecurity.” In *Handbook of Statistics, Vol. 35*, pp. 117–167. (Elsevier).

[6] Afzali Seresht, N., et al. (2020). “Investigating cyber alerts with graph-based analytics and narrative visualization.” *24th Int’l Conf. Information Visualization (IV)*. (Demonstrates a knowledge-graph and storytelling approach to alert analysis, supporting the efficacy of narrative techniques in SOCs.)